Unit Testing Framework for Operating System Kernels

Walter, Maxwell; Karlsson, Sven

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

Unit Testing Framework for Operating System Kernels

Maxwell Walter*, Sven Karlsson†
Technical University of Denmark

1 Introduction

Testing is an integral part of the development process, and many user-space applications and libraries have been designed to support system testing and analysis. Operating systems, however, present a unique challenge as the usual infrastructure required for testing is either not present or is the subject being tested.

To address the difficulty of testing a new operating system as it is being created, we are developing a testing framework in conjunction with our research operating system, FenixOS. By developing the testing infrastructure in conjunction with the operating system kernel itself, we ensure that testing is a fundamental aspect of the system design. In order to gain insight into kernel operation and state during runtime, we use virtual machine introspection, VMI, a technique typically used for intrusion detection [2, 4].

This poster presents the testing framework that we have developed, which consists of a test management and monitoring application utilizing VMI, and an in-kernel testing API. We are using this framework to assist in the development of FenixOS by providing unit tests during development, as well as automated integration testing to detect breakage. It also allows us to produce adversarial and cooperative tests, which require knowledge of the internal state of the kernel.

2 Testing Framework

Testing requires information about the application or process that is running in order to perform analysis and to report on test status. This means that some method of communication with an external observer is needed. This is more difficult in the case of an operating system kernel as the kernel typically provides this communication infrastructure. To obtain the information required, we run the kernel in a virtualization environment, and use introspection to view the kernel's internal state.

The virtualization environment we use as the basis of our test platform is QEMU. In addition to being a virtualization platform, QEMU is able to act as a hypervisor using the Linux KVM (kernel virtual machine) module [1]. This, combined with PCI and USB pass-through, allows for nearly bare-metal execution from the operating system's perspective and provides a more accurate testing environment while still enabling advanced analysis.

The communication methods provided by the introspection platform are split into two types; explicit and implicit. Explicit communication is when the kernel intentionally transmits information, and implicit communication is acquired by the framework without intentional action on the part of the kernel. Inside the kernel we have defined an API that provides a common base for implementing tests. The API communicates explicitly with the framework for reporting, and implicitly by providing test points in the form of static data structures. The in-kernel API is based on xUnit, which is a well known framework for describing and performing unit testing [3].

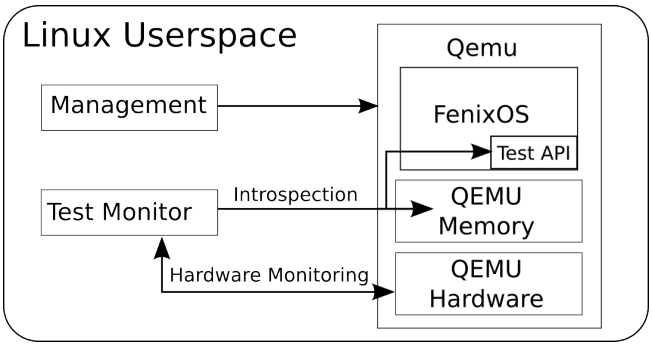

Figure 1: Test framework, showing management and monitoring components.

3 Conclusions

We are using the framework to aid in the development of FenixOS, which is currently in early stages of development. The management application provides us with a scriptable interface for testing multiple kernels built with different parameters, as well as different hardware configurations such as memory size. We have also implemented a number of tests during the development of both the memory management system and a minimal c++ kernel runtime. These components were developed independantly by different individuals, and having these tests and the framework to run them helped prevent code divergence as it was immediately evident when changes broke other parts of the system.

References

[1] BELLARD, F. Qemu, a fast and portable dynamic translator. *Proceedings of the USENIX 2005 Annual Technical Conference* (2005).

[2] GARINKEL, T., AND ROSENBLUM, M. A virtual machine introspection based architecture for intrusion detection. *Proceedings of Network and Distributed Systems Security Symposium* (2003).

[3] HAMILL, P. *Unit Test Frameworks, chapter Chapter 3: The xUnit Family of Unit Test Frameworks*. O'Reilly, 2004.

[4] NANCE, K., HAY, B., AND BISHOP, M. Virtual machine introspection. *IEEE Computer Society* (2008).

*Student. e-mail:maxw@dtu.dk
†svea@dtu.dk